ESP — A PACKAGE FOR THE EVALUATION OF STOCHASTIC PETRI NETS WITH PHASE-TYPE DISTRIBUTED TRANSITION TIMES

Aldo Cumani

Reprinted from PROCEEDINGS OF THE INTERNATIONAL WORKSHOP ON TIMED PETRI NETS, Torino, Italy, July 1-3, 1985

ESP - A PACKAGE FOR THE EVALUATION OF STOCHASTIC PETRI NETS WITH PHASE-TYPE DISTRIBUTED TRANSITION TIMES

Aldo Cumani

Istituto Elettrotecnico Nazionale Galileo Ferraris
Strada delle Cacce, 91 - 10135 Torino, Italy

Abstract - This paper describes a software package for the analysis of stochastic Petri net models. The package allows transition firing times to be distributed as Phase type (PH), i.e. as the time to absorption of continuous time, finite state Markov chains with at least one absorbing state. The PH class includes families of distributions commonly used in stochastic modelling, such as the Erlang, the hyperexponential, and the exponential distribution. The package provides facilities for entering the net description and the transition models, and for defining measures of system performance to be evaluated at the solution step. Both steady-state and transient solutions can be obtained; a Courtois-type aggregation technique is used in presence of stiff equations. The package is of conversational type and is written in FORTRAN 77; it has been developed on a PDP 11/44 under RSX11M, but has also been tested under VAX/VMS and VAX/UNIX.

1. INTRODUCTION

Petri nets (PN) and other similar graphical models, with the addition of some deterministic or stochastic timing mechanism, are becoming increasingly popular as tools for analyzing the performance of systems that exhibit concurrent and/or conflicting behaviours. The standard Stochastic PN's [1,2] are probably the best known example of such models in the stochastic case; they, however, restrict the class of firing time distributions to the negative exponential, that in many cases is a quite crude approximation of reality.

On the other hand, the introduction of general firing time distributions in Petri nets raises non-trivial problems with the precise definition of firing policies; different policies may yield different stochastic processes for the same net, sometimes showing behaviours far away from the modeller's intentions. A first extension of SPN's to the non-exponential case was indeed proposed by Natkin [1], leading to a semi-Markov process that is not realistic in the case of concurrent (parallel) activities [3].

To the author's knowledge, the only solution to the above problems that still yields an analytically tractable model, requires to restrict the class of allowed distributions to those of the phase type (PH) [4]. Attempts to extend the class of distributions to those of the phase type by using net-level constructs have already been reported in the literature; they, however, seem not easily mechanizable and have the drawback of introducing spurious items (places and transitions) in the model. On the other hand, the introduction of PH distributions at the reachability tree level is easily done, leading to a representation of system behaviour by a Markov chain as for SPN's [5].

A formal justification of this approach is beyond the scope of this paper; an extensive discussion of the problem of associating firing times of general distribution to Petri nets can be found in [5].

This paper describes an interactive software package (ESP: Evaluator of Stochastic Petri nets of Phase type), that allows to analyze a Petri net model with transition firing times following general PH distributions, and with several choices for the firing policy. The time behaviour of the model is evaluated by building an expanded Markov chain from the net reachability graph and the firing time PH models; the expanded chain is then solved for either the steady-state or the transient behaviour. Net structure, time models and rate parameters are entered using suitable symbolic languages; the marking-dependency of timing parameters may be of general form. The user may also enter definitions of system performance measures to be evaluated at the solution step.

The paper is organized as follows: in Sec. 2 we report some basic definitions and the algorithm for building the expanded Markov chain; Sec. 3 describes the ESP package in some detail, while Sec. 4 reports an illustrative example.

2. PETRI NETS AND PH DISTRIBUTIONS

We refer the reader to [5] for a discussion of theoretical issues related to Petri nets with firing times of general distribution; we only give here some basic definitions in order to make the paper as self-consistent as possible. Knowledge of the previously quoted references on stochastic Petri nets shall also be useful to the reader. We shall also refer to [6] for the generalized models (GSPN) including 'immediate' (i.e., zero time) transitions.

2.1 - Petri nets

The concept of Petri net is assumed known [7]. For our purposes, a marked PN is a 5-tuple (P,T,I,O,M1) where:

- P is the set of places, |P|=np
- T is the set of transitions, |T|=nt
- I c PxT is the set of transition input arcs
- O c TxP is the set of transition output arcs
- M1 is the initial marking of the net

In general, a marking of the net is an np-vector M=[m1,m2...mnp] where mi is the number of tokens in place pi. Input and output mappings are more conveniently represented by the ntxnp matrices DI and DO, where dIjk is the number of input arcs to transition tj from place pk, and similarly dOjk is the number of output arcs from transition tj to place pk. The enabling of transition tj in marking M is represented by the condition

$$mk \geq dIjk \ , \ k=1,2...np \qquad (2.1)$$

Firing an enabled transition tj changes the marking from M to M', denoted M-tj->M', where

$$m'k = mk + dOjk - dIjk \ , \ k=1,2...np \qquad (2.2)$$

The reachability set R(M1) of the PN is the set of markings that can be generated from M1 by repeated application of the firing operation. The reachability graph G(M1) of the PN is the labeled directed graph on R(M1) having an arc (Mi,Mj) with label tk for each triple (i,j,k) such that Mi-tk->Mj.

2.2 - Transition firing times

Several choices are possible for introducing time in Petri net models, depending upon the intended model semantics. We choose to associate random firing times with net transitions, in the sense that a transition fires after a given delay from the instant at which it became enabled; furthermore, this delay is a continuous random variable following a phase type distribution. In this way, the PN can be wieved as a compact graphical model of a continuous time, finite state stochastic system. Different net markings represent distinct system states; an enabled transition indicates that an activity associated to that transition is being carried on, and shall end up when the transition fires.

It must be observed that the above assumption does not uniquely determine the net behaviour in time; we must still specify:

1) which of several simultaneosly enabled transitions actually fires, and

2) what happens to transitions that have been enabled for some time, when some other transition fires, so changing the system state (net marking).

As problem 1) is concerned, our package only admits a race-type firing policy: the enabling of a transition represents the start of the corresponding activity, and the transition that actually fires is the one whose activity completes first. It must be remarked that the present version of our package does not support immediate transitions as in GSPN's, i.e. transitions that have higher priority than timed ones and fire in zero time upon their enabling. A transition priority structure is however provided, and immediate transitions can be simulated by high-rate, high-priority timed ones, later discarding 'vanishing' markings that enable them by aggregation of the resulting Markov chain (see Sec. 2.5). In this way, we can also simulate a preselection policy [5] where the transition to be fired is selected independently of its duration.

As problem 2) is concerned, there are still several possible choices. Our package admits the following three cases:

- resampling policy: each time a transition fires, the activities carried over by other enabled transitions are aborted, so when they become enabled again their durations are resampled from the corresponding distributions;

- age memory policy: in this case activities that did not complete are not aborted, but only suspended; when they become enabled again, they resume from the point of suspension;

- enabling memory policy: a transition following this policy is reset to its initial state when the change of marking disables it, while keeping memory of the work done if it is still enabled in the new marking (this construct is useful to model conflicting activities whose completion disables each other, while they are not affected by the completion of some parallel activity carried over in some other portion of the system).

2.3 - PH distributions

Phase type (PH) distributions are defined [4] as those of the time till absorption in finite state Markov chains (FSMC) with at least one absorbing state. The class of continuous time PH includes several families of distributions commonly used in applied stochastic modelling, such as the exponential, Erlang, and hyperexponential.

For our purposes, the distinct advantage of PH distributions is their representation with FSMC's; this means that each activity is modelled by a transition graph whose nodes represent the 'stage of firing' reached by the transition in the course of time. In this way, the stochastic point process representing the net behaviour in time is itself a homogeneous Markov process [5], whose states are identified by the net marking and the stage of firing of each transition; its transition graph can be constructed expanding the reachability graph G(M1) by a simple enumerative algorithm [8], that can easily accommodate the various firing policies previously defined.

2.4 - The expansion algorithm

A brief sketch of the algorithm used to build

he expanded Markov chain is discussed in this ubsection. We assume that for each transition ti PH model has been specified, having ni states, ith a single initial state numbered 1 and a single inal state numbered ni.

The expanded process is represented by a arkov transition graph H=(NH,AH) where NH is the et of nodes (Markov states) and AH is the set of cs. The nodes in NH are pairs (M,W), where M is marking in R(M1) and W is an integer nt-vector ose i-th entry wi (1≤wi<ni) represents the stage firing of ti, i.e. the node number in the graph its PH model.

Arcs in AH are represented by 5-tuples (N,N';i,j,k) where N is the initial node, N' the final node and (j,k) is an arc in the PH model of transition ti; therefore, (N,N';i,j,k) ∈ AH indicates that the process goes from N to N' when the stage of firing of ti changes from j to k.

The expansion algorithm is shown, in Pascal-like form, in Fig. 1. The expanded graph is initially empty; since we suppose that at the beginning (t=0) all system activities start anew, we put in NH the node (M1,[1,1...1]) and mark it as non-expanded.

An expansion step is then performed on each non-expanded node N=(M,W). For each transition ti enabled in M, its PH model is searched for Markov

```
(* algorithm to build the expanded Markov graph *)

begin       (* initialize *)

 AH := ∅ ; W=[1,1,...1]; N: = (M1,W) ; NH := { N } ;
 "mark N as non-expanded" ;

 (* expansion loop *)

 repeat
   begin (* constructing all successor nodes of N *)
   for i := 1 to nt do
     if (ti is enabled) then
       for k := 1 to ni do
         if "(w[i],k) is an arc in PH model of ti" then
           begin (* construct a successor node *)
             for j := 1 to nt do
               if j = i then
                 w'[j] := k
               else
                 w'[j] := w[j] ;
             N' := (M,W') ;
             if k = ni then
               begin (* ti has fired *)
                 "get the new marking M': M-ti->M'"
                 for j := 1 to nt do
                   if j = i then
                     w'[j] := 1
                   else
                     (* set w'j according to firing rules *)
                     case "memory policy of tj" of
                       resampling : w'[j] := 1  ;
                       age        : w'[j] := w[j] ;
                       enabling   : if (tj is enabled in M') then
                                      w'[j] := w[j]
                                    else
                                      w'[j] := 1 ;
                     end (* of case *)
                 N' := (M',W');
               end
           end (* of construction of a successor node*)
           if N' ∉ NH then
             NH := NH U {N'} (* store N' as a new Markov state *)
             "mark N' as non-expanded" ;
           AH := AH U { N-(ti,wi,k)->N' } (* add the arc N->N' to AH *)
   end (* of construction of successor nodes of N *)
 "mark N as an expanded node"
 N := "next non-expanded node in NH"
 until (no more nodes to be expanded in NH)

end.
```

. 1 - The algorithm to build the expanded process, in Pascal-like form

transitions outgoing from stage wi. Then, for each k such that there is an arc (wi,k) in the model of ti, a possible successor node N'=(M,W') is generated, with w'i=k and w'j=wj for j≠i.

Now, if k≠ni, (M,W') is a true Markov state; while if k=ni transition ti has fired, and the successor node is renamed as (M',W") with M-ti->M' and w"i=1 since the completion of ti resets it to the initial condition. The entries of W" corresponding to other enabled transitions shall be set according to the chosen memory policy (i.e., always reset in the resampling case, not reset in the age case, conditionally reset in the enabling case). In both cases, the successor node is entered in NH (if not yet there), and an arc (N,N';i,wi,w'i) is added to AH. This procedure is then repeated until there are no more non-expanded nodes in NH.

The exact number Ns of states in the expanded process depends on both the net structure and the firing policy; in the resampling (R) case we have:

$$Ns(R) = \sum_{M \in R(M1)} \prod_{tj \in E''(M)} (ni-1) \qquad (2.3)$$

where E"(M) is the set of transitions enabled in M; in the enabling (E) and age (A) memory cases, Ns is not easy to compute without actually building the expanded graph; an upper bound is however

$$Ns(E) \leq Ns(A) \leq N \prod_{tj \in T} (ni-1) \qquad (2.4)$$

where N = |R(M1)|.

It should be remarked again that introducing the PH representation at the reachability graph level has several advantages over the use of net-level constructs as proposed in the literature [2]. First, as we explained above, the analysis procedure (from the PN to the Markov chain) is easily mechanized while still taking into account several different firing policies. On the contrary, expanding at the net level seems not so easy to do, since the implementation of the various policies may require the introduction of additional items (places, transitions) to properly link together the subnets representing the PH expansion of the original transitions.

Second, but not less important, the addition of places and transitions that do not refer to the actual system operation, but only to the timing of events, makes the expanded net clumsy and difficult to understand. In the resulting reachability graph, it is not easy to recognize which markings correspond to a particular state of the original, non-expanded model; on the contrary, in our approach the net and its reachability set are not modified, while the sets of Markov states corresponding to some non-expanded state can be easily identified.

2.5 - Aggregation

The numerical solution of the expanded chain, particularly in the transient case, is a non-trivial problem when the transition rate matrix is stiff (i.e. with rate values differing by orders of magnitude).

Stiffness cannot be avoided in most practical cases, and in particular when simulating immediate transitions by high-rate timed ones; in these cases, however, an approximate solution can be obtained by aggregation of the expanded process. We give here only a brief sketch of the aggregation algorithm implemented in ESP; a more detailed discussion may be found in [9].

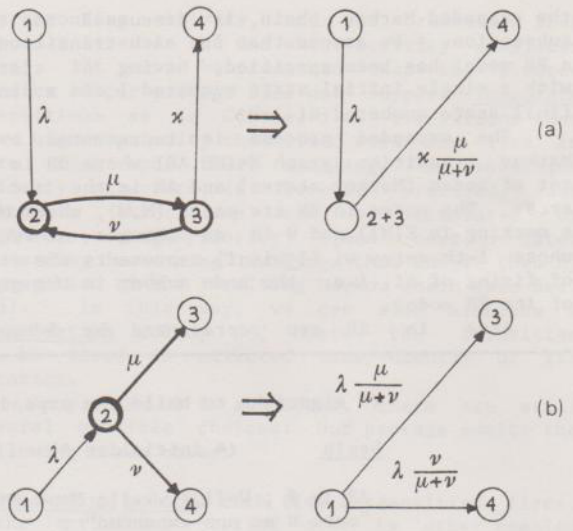

Fig. 2 - Aggregation : a) a set of recurrent fast states (bold); b) a transient fast state (bold). Labels on arcs are rate values.

Markov states are first classified as fast or slow by comparing the rate values of outgoing arcs in the graph with a user-defined threshold. Fast states are then labeled as transient or recurrent according to the graph obtained by deleting slow arcs (recurrent states are those that, after this deletion, belong to an ergodic set). Each set of recurrent states is merged into a single dummy slow state [10], and the rate values of arcs steaming from states of the set are modified according to the equilibrium probability distribution among the same states (refer to Fig. 2a for a simple example).

Transient states, on the other hand, are discarded by suitably rearranging the connections between slow states (Fig. 2b).

From the solution of the aggregate process, the state probabilities of the original process can then be recovered by suitably weighting the aggregate solution.

It must be noticed that the computation of aggregation/disaggregation weights requires the steady-state probabilities of recurrent states; ESP allows either the use of approximate steady-state probabilities computed, for each set of recurrent states, by discarding slow arcs from the graph, or of the exact values obtained from the solution of the whole process. The latter choice has the advantage of ensuring an asymptotically exact solution; on the other hand, the first choice is computationally more efficient, and may also be used to get an approximate steady-state solution (e.g., for very large problems).

3. THE ESP PACKAGE

ESP is implemented as an interactive program composed of a command interpreter module that reads directives from a video terminal or a disk file, and calls one of several other modules that do the actual computations (e.g. construction of the reachability graph, evaluation of the steady-state solution etc.); refer to Table I for a list of available directives. Intermediate results are put on disk files, so keeping main memory requirements at a minimum; this modular structure allows ESP to run even on small (16 bit) machines.

ESP has been developed on a PDP 11/44 under RSX11M. It is entirely written in standard FORTRAN 77 and is therefore easily portable to other systems; it has already been tested under VAX/VMS and UNIX. Input data to ESP are given using a free-format symbolic language that makes quite easy to enter and change model specifications.

As can be seen from Table I, the present version of ESP allows the user to:

1) enter the PN description;
2) build the corresponding reachability graph;
3) enter PH model structures for the transition firing times;
4) build the expanded Markov transition graph of the process;
5) specify parameter values for the PH models, and performance measures to be computed from the process solution;
6) solve the expanded process and compute performance measures for both the equilibrium solution and the transient solution;
7) build an aggregate version of the expanded graph in order to overcome stiffness problems in the transient solution step.

Steps 1)-6) correspond, in that order, to the 'natural' sequence of steps needed to get an evaluation of system performance from the model, however it should be clear that the user may reenter this sequence at any step, e.g. to change some parameter value and see how this change affects the final results. Step 7) may result very useful when dealing with stiff problems (i.e., when transition durations differ by orders of magnitude).

In the following, each of the above steps is considered in some detail. In addition to the above, ESP provides facilities for displaying results in either tabular form or, where appropriate (e.g. for the reachability graph), in graphical form. It also provides a simple line-oriented text editor for entering and modifying problem specification data; the RSX11 version allows direct calls to the more powerful screen editor EDT. An HELP facility is provided in order to assist the user with explanations about program directives and input data syntax.

3.1 - PN description and construction of the reachability graph

The net description is entered as a sequence of statements that assign unique names to places and transitions, specify the input and output places to each transition, assign the initial

BAT	direct input to a batch file (a file of ESP directives)
BLOW	build the expanded Markov graph
COMP	compile definitions of rate parameters and performance measures
DRAW	display data/results in graph form
EDIT	edit problem data files
EQUI	compute the equilibrium (steady-state) solution
GERT	generate the reachability tree
HELP	display help information
LMAC	list defined macros
LUMP	build an aggregate version of the process
MAC	define a macro (sequence of directives)
OUT	direct ESP output to a listing file
PURG	delete a macro
SET	set directive switches
SHOW	show work file names & switch status
TRAN	compute the transient solution
TYPE	display data/results in tabular form
VALM	compute the expanded transition matrix
USE	define work file names
XEQ	execute a macro

Table I - List of available ESP directives

name	C	meaning
TOKENS	1	number of tokens in a set of places
ETOK	2	expected number of tokens in a set of places
AV.ETOK	2	time average of ETOK
EFIRE	2	expected number of firings per unit time of a set of transitions
AV.EFIRE	2	time average of EFIRE
PR	2	probability of a set of markings
AV.PR	2	time average of PR

Table II - List of special ESP-defined functions. C: 1 = may be used to define rate parameters; 2 = may be used to define output measures.

marking to places and (optionally) priorities to transitions. Inhibitory arcs may be entered by flagging with a minus sign the place name in the specification of transition inputs. An example of a PN description file is reported in Sec. 4.

From the net description, the program builds the reachability graph by enumerating reachable markings obtained by application of eq.(2.2). The program also checks whether the reachability set is finite or not, and, if required, makes a partial reduction of the graph by discarding markings that enable only one high-priority transition.

3.2 - PH model description and construction of the expanded graph

Similarly to the net description, the specification of PH models of firing delays is accomplished by a set of statements that, for each

transition, assign a model structure and a memory policy chosen among the three allowed cases (namely, resampling, age or enabling). Arbitrary PH models may be defined by entering the order (number of Markov states minus one) and the set of arcs (from state, to state) of the corresponding Markov transition graph.

For ease of data entry, however, the graph structures for commonly used subclasses such as exponential, Erlang and hyperexponential are predefined in the program (the hyperexponential is represented by a two-stage model) and are recalled by the keywords /EXP, /ERL and /HYP ; refer to Sec. 4 for an example of model specification file.

From the reachability graph and the PH models, the expanded Markov transition graph is built by the algorithm of Sec. 2.4. Markov states are arranged in a lexicographic ordering, so a dichotomic search algorithm can be used to speed up the search for newly generated states.

3.3 - Specification of input parameters and output measures

The numerical values of the PH model parameters are defined by a set of statements written in a simple functional language, that allows them to be specified by arbitrary algebraic expressions involving constants, other similarly defined parameters, and system-supplied or user-defined functions. This allows an easy definition of marking-dependent transition rates; for example, a statement such as

```
R.T1.1.3 = 9-3*TOKENS:P2;
```

defines the rate of arc (1,3) in the model of transition "T1" as being equal to 9 minus three times the number of tokens in place "P2" (TOKENS is an ESP-supplied function having a place name as argument). More complicated marking dependencies may also be accommodated by using program-supplied numerical functions (e.g. LOG, EXP, SIN etc.) and conditional (IF..THEN..ELSE..ENDIF) expressions. For ease of data entry, the parameters for EXP, ERL and HYP distributions may be fully specified by assigning only the mean value (LIFE) and, for HYP, the squared coefficient of variation (CV2).

Besides rate parameters, items to be evaluated and typed out in the solution phase (e.g., performance measures) may also be defined in terms of marking probabilities and of their time averages, either explicitly or through ESP-defined functions that directly yield the expected number of tokens in a place and the expected rate of firing of a transition. The list of available functions is reported in Table II.

The specification statements are processed by a compiler module and translated into a pseudo-code that can be accepted by an interpreter; this latter is called by the module that computes the transition rate matrix of the expanded process, and, if performance measure computations have been requested, by the modules that solve the process.

3.4 - Steady-state and transient solutions

The steady-state solution of the expanded process, i.e. the solution of

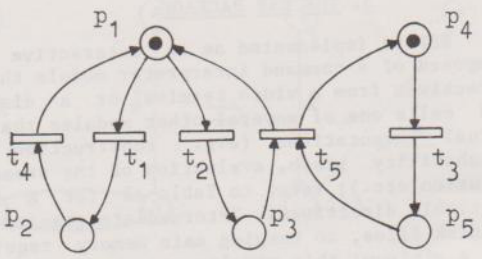

Fig. 3 - The net used in the example of Sec. 4

$$A\,P = 0 \qquad (3.1)$$

where A is the expanded transition rate matrix and P the vector of Markov state probabilities, is accomplished by a standard, sparse matrix, Gauss elimination algorithm. The solution module automatically aggregates the results according to the net marking/Markov state mapping, so yielding the solution also for marking probabilities.

The transient solution of the process is

$$P(t) = \exp(At)\,P(0) \qquad (3.2)$$

and is computed recursively at equispaced sampling times as

$$P(t+s) = \exp(As)\,P(t) \qquad (3.3)$$

where s is the sampling interval.

Eq. (3.3) is evaluated [11] by series expansion of the right hand side. This approach has the advantages of ease of programming and of small memory requirements (the successive terms in the r.h.s. of (3.3) can be computed recursively, without actually computing powers of the matrix As); also, due to the properties of transition rate matrices, usually few terms of the series are required to get the desired accuracy. This method breaks down in the case of stiff problems, but for that case an approximate solution can still be obtained by aggregation of the expanded process (Sec. 2.5).

4. A SAMPLE ESP SESSION

In order to illustrate how ESP works, we report here a sample session referring to the analysis of the net of Fig. 3. This is quite a simple net, yet it exhibits both conflict (t1,t2) and parallelism (t3), and is also bounded and live. Such a net is described to ESP by the file EXAMPLE.NET that contains the following data:

```
PLACE P1 P2 P3 P4 P5
TRANS T1 T2 T3 T4 T5
IN T1 P1 OUT T1 P2
IN T2 P1 OUT T2 P3
IN T3 P4 OUT T3 P5
IN T4 P2 OUT T4 P1
IN T5 P3 P5 OUT T5 P1 P4
MARK P1 1
MARK P4 1
```

Assuming for the firing times of t1 and t2 an Erlang distribution of order 2, with enabling memory policy, the PH model specification file looks like that:

```
MODEL T1 /ERL 2 /ENABLING
MODEL T2 /ERL 2 /ENABLING
```

(transitions without an explicit model declaration are assumed exponential by default). The rate parameters of the distributions are defined by:

```
LIFE.T1=1; LIFE.T2=1; LIFE.T3=3-2*TOKENS:P1;
LIFE.T4=1; LIFE.T5=1; LIFE.T6=1;
GROUP:FIRSTTWO=P1,P2;
$ MEASURE=ETOK:FIRSTTWO;
```

that assign the mean values (LIFE); notice that T3 is marking-dependent. The above statements also define an item (MEASURE) to be evaluated at the solution step (flagged by the dollar sign), and equal to the expected number of tokens (ETOK) in the set of places (P1,P2) named by the GROUP statement.

Fig. 4 shows an example session using the above data. Directives are entered by the user after the prompt "ESP>". The USE directive defines the names of input and work data files; GERT then requests the computation of the reachability set R(M1) and graph G(M1). DRAW RTR then prints a (semi)graphic drawing of the reachability graph, and BLOW builds the expanded Markov graph from G(M1) and the PH models defined above.

The rate specification file is then compiled by COMP; VALM then uses such specifications to compute the transition rate matrix, that is typed out by TYPE VAL. The steady-state solution is then computed by EQUI, that types out the values of Markov state and marking probabilities, and the value of the output measure MEASURE previously defined. Finally, TRAN computes the transient behaviour of that measure in the time range (0,1) at intervals of 0.2.

5. CONCLUSIONS

The package presented in this paper is intended to provide a workable tool for system performance evaluation. The availability of a wide class of distributions to model system activity durations, together with the accommodation of different firing policies, considerably extends the modelling power of the system representation with respect to the all-exponential assumption of standard SPN's.

Several improvements and extensions are obviously possible; for example, ESP does not yet support immediate transitions as in GSPN's; furthermore, a unique language for describing the net, the PH models and the parameter specifications could be desirable.

It should be remarked that, in spite of the relative user-friendliness of ESP, its full capabilities are probably not within the reach of naive users, due to the subtle problems arising in connection with general firing time distributions. On the other hand, experienced users may get better answers to their problems by using distributions that fit more closely the real system behaviour.

ACKNOWLEDGMENT

The author wishes to thank the co-authors of paper [5], and particularly A. Bobbio, without whose incitement this work would probably have never been undertaken.

REFERENCES

[1] S. Natkin: "Les reseaux de Petri stochastiques et leur application a l'evaluation des systemes informatiques", These de Doct. Ing., CNAM, Paris (1980)

[2] M. K. Molloy: "Performance analysis using stochastic Petri nets", IEEE Trans Comput, C-31, 9, 913-917 (1982)

[3] J. B. Dugan, K. S. Trivedi, R. Geist, V. F. Nicola: "Extended stochastic Petri nets: applications and analysis", Performance 84 (1984)

[4] M. Neuts: "Matrix-geometric solutions in stochastic models", Johns Hopkins Univ. Press (1981)

[5] M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte, A. Cumani: "On Petri nets with stochastic timing", Int Workshop Timed Petri Nets, Torino (1985)

[6] M. Ajmone Marsan, G. Balbo, G. Conte: "A class of generalized stochastic Petri nets for the performance analysis of multiprocessor systems", ACM TOCS, vol 2, n 2 (1984)

[7] J. L. Peterson: "Petri net theory and the modeling of systems", Prentice Hall (1981)

[8] A. Bobbio, A. Cumani: "Discrete state stochastic systems with phase-type distributed transition times", AMSE Int Conf Modelling & Simulation, Athens (1984)

[9] A. Bobbio, A. Cumani, R. Del Bello: "Reduced markovian representation of stochastic Petri net models", Systems Science, vol 10, n 2 (1984)

[10] P. J. Courtois, "Decomposability: Queuing and Computer System Applications", Academic Press (1977)

[11] M. Ajmone Marsan, A. Bobbio, G. Conte, A. Cumani: "Performance analysis of degradable multiprocessor systems using generalized Petri nets", IEEE Comp Soc DPTC Newsletter, 6, SI-1 (1984)

```
ESP>USE ALL EXAMPLE
ESP>GERT
Priorities shall not be used
RTREE IER=     0
NSTA=     6 NARC=    10 LUSED=   125

  STA PRIO MARKING
          P1  P2  P3  P4  P5
   1   0   1   0   0   1   0
   2   0   0   1   0   1   0
   3   0   0   0   1   1   0
   4   0   1   0   0   0   1
   5   0   0   1   0   0   1
   6   0   0   0   1   0   1

REACHABILITY GRAPH:
 FRS  TOS TRA TOS TRA TOS TRA
   1   2  T1   3  T2   4  T3
   2   5  T3   1  T4
   3   6  T3
   4   5  T1   6  T2
   5   4  T4
   6   1  T5

ESP>DRAW RTR

Reachability tree:

  +--+  1+--+  3+--+  4+--+  1+--+
  | 1|-->| 2|-->| 5|-->| 4|-->| 5|
  +--+   +--+   +--+   +--+   +--+
    |      |             |
    |      |             |   2+--+  5+--+
    |      |             +-->| 6|-->| 1|
    |      |                 +--+   +--+
    |      |
    |      |   4+--+
    |      +-->| 1|
    |          +--+
    |
    |   2+--+  3+--+
    +-->| 3|-->| 6|
    |   +--+   +--+
    |
    |   3+--+
    +-->| 4|
        +--+

ESP>BLOW
#### COMPUTING THE EXPANDED MARKOV GRAPH ####
Number of markov states is    12 NARC=   25

 STA MARK  PH state
         T1  T2  T3  T4  T5
   1   1   1   1   1   1   1
   2   1   2   1   1   1   1
   3   1   1   2   1   1   1
   4   4   1   1   1   1   1
   5   2   1   1   1   1   1
   6   1   2   2   1   1   1
   7   4   2   1   1   1   1
   8   3   1   1   1   1   1
   9   4   1   2   1   1   1
  10   5   1   1   1   1   1
  11   4   2   2   1   1   1
  12   6   1   1   1   1   1
```

```
(continued)
TRANS MAT:

FRS  TOS RID TID TOS RID TID TOS RID TID
  1    2   1   0   3   3   0   4   5   3
  2    5   2   1   6   3   0   7   5   3
  3    6   1   0   8   4   2   9   5   3
  4    7   1   0   9   3   0
  5   10   5   3   1   6   4
  6    5   2   1   8   4   2  11   5   3
  7   10   2   1  11   3   0
  8   12   5   3
  9   11   1   0  12   4   2
 10    4   6   4
 11   10   2   1  12   4   2
 12    1   7   5

ESP>COMP/NOLIST
ESP>VALM
ESP>TYPE VAL
File type is      1
Num. states=   12
        1       2       3       4       5       6       7       8       9      10      11      12
  1-5.E+00   ...     ...     ...   1.E+00   ...     ...     ...     ...     ...     ...   1.E+00
  2 2.E+00-5.E+00   ...     ...     ...     ...     ...     ...     ...     ...     ...     ...
  3 2.E+00   ... -5.E+00   ...     ...     ...     ...     ...     ...     ...     ...     ...
  4 1.E+00   ...     ... -4.E+00   ...     ...     ...     ...     ...   1.E+00   ...     ...
  5   ...   2.E+00   ...     ... -1.E+00 2.E+00   ...     ...     ...     ...     ...     ...
  6   ...   2.E+00 2.E+00   ...     ... -5.E+00   ...     ...     ...     ...     ...     ...
  7   ...   1.E+00   ...   2.E+00   ...     ... -4.E+00   ...     ...     ...     ...     ...
  8   ...     ...   2.E+00   ...     ...   2.E+00   ... -3.E-01   ...     ...     ...     ...
  9   ...     ...   1.E+00 2.E+00   ...     ...     ...     ... -4.E+00   ...     ...     ...
 10   ...     ...     ...     ...   3.E-01   ...   2.E+00   ...     ... -1.E+00 2.E+00   ...
 11   ...     ...     ...     ...     ...   1.E+00 2.E+00   ...   2.E+00   ...  -4.E+00   ...
 12   ...     ...     ...     ...     ...     ...     ...   3.E-01 2.E+00   ...   2.E+00-1.E+00

ESP>EQUI
SOLEQO IER=    0  LUSED=   123 NGARB=   0
EQUILIBRIUM DISTRIBUTION (MARKOV STA):
  1  0.58616647D-01   2  0.23446659D-01   3  0.23446659D-01   4  0.56271981D-01
  5  0.63305979D-01   6  0.18757327D-01   7  0.33997655D-01   8  0.25322392D+00
  9  0.33997655D-01  10  0.16647128D+00  11  0.38686987D-01  12  0.22977726D+00
EQUILIBRIUM DISTRIBUTION (MARKINGS):
  1  0.12426729D+00   2  0.63305979D-01   3  0.25322392D+00   4  0.16295428D+00
  5  0.16647128D+00   6  0.22977726D+00
MEASURE                        =  5.169988276671D-01

ESP>TRAN/NOLIST
EPS?
TFIN,STEPSIZE?1.,.2
INITIAL STATE [Default=1]?

Integration:
step    1   time=   2.00000E-01   ncycl=   13
MEASURE                        =  9.407465305741D-01
step    2   time=   4.00000E-01   ncycl=   12
MEASURE                        =  8.310955761667D-01
step    3   time=   6.00000E-01   ncycl=   12
MEASURE                        =  7.318755501817D-01
step    4   time=   8.00000E-01   ncycl=   11
MEASURE                        =  6.610197058270D-01
step    5   time=   1.00000E+00   ncycl=   11
MEASURE                        =  6.159687837073D-01

ESP>EX
[exit]
```

Fig. 4 - A sample ESP session. User directives follow the prompt ESP>.